

UNITED STATES PATENT APPLICATION

HIGH-PERFORMANCE ADDER

INVENTORS

SANU K. MATHEW

of Hillsboro, OR, USA

RAM KRISHNAMURTHY

of Beaverton, OR, USA

Schwegman, Lundberg, Woessner, & Kluth, P.A.
1600 TCF Tower
121 South Eighth Street
Minneapolis, Minnesota 55402
ATTORNEY DOCKET 884.448US1
Client Ref. No. P11223

HIGH-PERFORMANCE ADDER

5

Field of the Invention

The present invention relates generally to adder circuits, and more particularly to a high performance, robust sparse carry-merge adder circuit.

Background Information

10

All processors, microprocessors or central processing units include arithmetic logic units (ALU) to perform calculations involving integers. An ALU will contain a multiplicity of adder circuits or simply adders to perform the arithmetic calculations by summing two binary numbers together. The binary numbers being added will typically contain either 32 bits or 64 bits. Accordingly, adders will typically be of two types of design or architecture, one to handle the summation of 32 bit binary numbers and another to handle the summation of 64 bit binary numbers.

15

20

Adders are generally used by the majority of instructions in controlling the operations of a computer system, microprocessor or the like and are performance limiting devices in such systems because they form a core of several critical paths in performing instructions and calculations. A typical adder circuit can include over 500 logic gates and will generate a carry for every two bits summed together for two 64 bit or 32 bit binary numbers. Accordingly, 64 summations and carries are generated in parallel operations. While the time period during which these arithmetic operations are performed is extremely fast, a simpler architecture with fewer gates would perform fewer operations in a much shorter time period. A larger number of logic gates and a longer period of time to perform operations will also result in more power consumption.

25

Additionally, the more complex adder circuit with hundreds of logic gates requires additional and more complex wiring to interconnect the hundreds of gates and will also occupy a considerable area or real estate on a semiconductor chip.

Another issue with some adders that use a multiplexer to select between different conditional sums or conditional carries is that a conditional sum or conditional carry input to the multiplexer may be erroneously discharged that could result in a later arriving true conditional sum input or true conditional carry input to the multiplexer not being evaluated correctly.

Accordingly, for the reason stated above, and for other reasons that will become apparent upon reading and understanding the present specification, there is a need for a robust, high performance adder with a simplified architecture, that occupies less area on a semiconductor chip and that operates faster and with less power consumption and is not subject to incorrect evaluation of conditional sums and conditional carries.

Brief Description of the Drawings

In the drawings, like numerals describe substantially similar components throughout the several views.

Figure 1 is a block schematic diagram of a 64 bit quaternary-tree adder in accordance with the present invention.

Figure 2 is a block schematic diagram of a sparse carry-merge circuit in accordance with the present invention.

Figure 3 is a block schematic diagram of an intermediate carry generator in accordance with the present invention.

Figure 4 is a block schematic diagram of a conditional sum generator in accordance with the present invention.

Figure 5 is a block schematic diagram illustrating the critical path of the 64 bit quaternary-tree adder of Figure 1.

Figure 6 is a block schematic diagram of a 32 bit quaternary-tree adder in accordance with the present invention.

Figure 7 is a schematic diagram of a multiplexer recovery circuit in accordance with the present invention.

Figure 8 is a flow chart of the method of operation of the quaternary-tree adder in accordance with the present invention.

Figure 9 is a block schematic diagram of an example of an electronic system that incorporates the quarternary-tree adder of the present invention.

Description of the Preferred Embodiments

In the following detailed description of the preferred embodiments, reference is made to the accompanying drawings which form a part hereof, and in which is shown by way of illustration specific embodiments in which the invention may be practiced. It is to be understood that other embodiments may be utilized and structural changes may be made without departing from the scope of the present invention.

Referring initially to Figure 1, a quarternary-tree adder 100 in accordance with the present invention is shown for summing two 64 bit (bits 0 to 63) binary numbers, A_{63} and B_{63} . The adder 100 includes a first circuit or a sparse carry-merge tree or circuit 102 into which the binary numbers, A_{63} and B_{63} , are inputted. The sparse carry-merge circuit 102 is adapted to generate a first predetermined number or group of carries or carry signals, C_{15} , C_{31} and C_{47} . The sparse carry-merge circuit 102 generates one (1) in sixteen (16) carries or will generate one (1) carry for each group of sixteen (16) bits of the 64 bit binary numbers grouping the bits together starting with a least significant digit or bit zero (0), A_0 and B_0 . Accordingly, for the 64 bit adder 100 the three carries C_{15} , C_{31} and C_{47} will be generated.

The sparse carry-merge circuit 102 is coupled to a plurality of second circuits or intermediate carry generators 104 adapted to generate a second predetermined number of carries or the missing one (1) in four (4) carries or carry signals from the first predetermined number or group of carries or carry signals. Accordingly, a first intermediate carry generator 104A will generate missing 1 in 4 carries C_3 , C_7 , and C_{11} , using a carry-in (C_{in}) which is passed through the first intermediate carry generator 104A. C_{in} can be a 0 or a 1. The second intermediate carry generator 104B will generate missing 1 in 4 carries C_{19} , C_{23} , and C_{27} using carry C_{15} from the first predetermined number of carries as a carry-in as will be described in more detail below. The third intermediate carry generator 104C will generate missing 1 in 4 carries C_{35} , C_{39} , and C_{43} using carry C_{31} as a carry-in, and the fourth intermediate carry generator

104D will generate missing 1 in 4 carries C_{51} , C_{55} and C_{59} using C_{47} as a carry-in. The intermediate carry generators 104, therefore, generate the additional carries or carry signals such that one carry is generated for every group of four bits of the two binary numbers, grouping the bits together beginning with the least significant digit or bit 0.

5 The adder 100 further includes a third circuit or plurality of sum generators 106 coupled to the intermediate sum generators 104 and the sparse carry-merge circuit 102 to provide a final sum 108 of the two binary number A_{63} and B_{63} . As will be described in more detail below, each sum generator 106 computes a sum for a group of four bits of the two binary number A_{63} and B_{63} using one of the one in four carries (or one of the first and second predetermined number of carries). For example, the first conditional sum generator 106A will compute a Sum(3:0) generated from bits 0-3 of the two binary numbers A_{63} and B_{63} using carry-in (C_{in}). Similarly, the second conditional sum generator 106B will compute a Sum(7:4) generated from bits 4-7 of the two binary numbers using carry C_3 as the carry-in and so forth for the other conditional sum generators 106 shown in Figure 1.

15 Referring also to Figure 2, which is a detailed block schematic diagram of the sparse carry-merge tree or circuit 102 in accordance with the present invention, the sparse carry-merge circuit 102 includes a first stage 200 to merge each of the bits 0-63 of the two binary numbers A_{63} and B_{63} together. The first stage 200 of the sparse carry-merge circuit 102 includes a plurality of logic gates 202 represented by the blocks in the first stage 200 to generate a plurality of propagate signals P_i (where $i=0, 63$) and a plurality of generate signals G_i (where $i=0, 63$). The logic gates 202 each perform an "exclusive OR" logic operation on each pair of the bits 0-63 of the binary numbers A_{63} and B_{63} to generate the plurality of propagate signals ($P_i=A_i \oplus B_i$) and the logic gates 202 also each perform an "AND" logic operation on each pair of bits of the two binary numbers A_{63} and B_{63} to generate the plurality of generate signals ($G_i=A_i \cdot B_i$).

20 The sparse carry-merge circuit 102 also includes a second stage 204 including a plurality of carry-merge (CM) logic gates 206 (represented by each of the blocks in the second stage 204). Each CM logic gate 206 is coupled to two adjacent logic gates 202 of the first stage 200 and will merge the two propagate signals (P_i and P_{i+1}) and the two

generate signals (G_i and G_{i-1}) from the two adjacent logic gates 202 of the first stage 200. The logic equation of the CM logic gate 206 for an output propagate signal is $P_{out} = P_i \cdot P_{i-1}$ and the logic equation of the CM logic gate 206 for an output generate signal is $G_{out} = G_i + P_i \cdot G_{i-1}$. Accordingly, the first CM logic gate 206A will merge the propagate signal P_0 and generate signal G_0 from the first logic gate 202A of the first stage 200 with the propagate signal P_1 and the generate signal G_1 of the second logic gate 202B of the stage 200. Similarly, the second CM logic gate 206B of the second stage 204 will merge the propagate signal P_2 and generate signal G_2 of the third logic gate 202C with the propagate signal P_3 and the generate signal G_3 of the fourth logic gate 202D of the first stage 200 and so forth for each of the CM logic gates 206 in the second stage 204.

A third stage 208 of the sparse carry-merge circuit 102 is coupled to the second stage 204 of the circuit 102. The third stage 208 includes a second plurality of CM logic gates 210. Similar to the relationship of the second stage 204 and the first stage 200, each CM gate 210 of the third stage 208 is coupled to two adjacent CM logic gates 206 of the second stage 204 to merge the two propagate signals (P_i and P_{i-1}) and the two generate signals (G_i and G_{i-1}) from the two adjacent CM logic gates 206 of the second stage 204.

The sparse carry-merge circuit 102 further includes a fourth stage 212, a fifth stage 214 coupled to the fourth stage 212 and a sixth stage 216 coupled to the fifth stage 214 to generate the 1 in 16 carries C_{15} , C_{31} and C_{47} for the two binary numbers A_{63} and B_{63} . The fourth stage 212 includes a plurality of CM logic gates 218. Each CM logic gates 218 of the fourth stage 212 is coupled to two adjacent CM logic gates 210 of the third stage 208 and merges the two propagate signals (P_i and P_{i-1}) and the two generate signals (G_i and G_{i-1}) from the two adjacent CM logic gates 210. Thus the CM gate 218A will merge the signals from the CM gates 210A and 210B, the CM gate 218B will merge signals from the CM gates 210C and 210D, and so forth for the other CM gates 218 of the fourth stage 212. Similarly, the fifth stage 214 includes a plurality of CM gates 220 that are each coupled to two adjacent CM gates 218 of the fourth stage 212 for merging the propagate signals (P_i and P_{i-1}) and the generate signals (G_i and G_{i-1}) from each pair of adjacent CM gates 218 of the fourth stage 212. The sixth stage 216

includes an inverter 222 and two CM gates 224A and 224B. The inverter 222 is coupled to the first CM gate 220A of the fifth stage 214. The output signal of the inverter 222 provides the first 1 in 16 carry C_{15} of the first predetermined number of carries. The first CM gate 224A of the sixth stage 216 is coupled to the first CM gate 220A and the second CM gate 220B of the fifth stage 214. The first CM gate 224A merges the propagate signal P_i and the generate signal G_i from the second CM gate 220B of the fifth stage 214 with the generate signal G_{i-1} from the first CM gate 220A of the fifth stage 214 to provide the second 1 in 16 carry C_{31} of the first predetermined number of carries. The second CM gate 224B of the sixth stage 216 is coupled to the first CM gate 220A, the second CM gate 220B and the third CM gate 220C of the fifth stage 214. Accordingly, the second CM gate 224B merges the propagate signal P_i and the generate signal G_i from the third CM gate 220C with the generate signal G_{i-1} from the second CM gate 220B and the generate signal G_{i-2} from the first CM gate 220A of the fifth stage 214 to provide the third 1 in 16 carry C_{47} of the first predetermined number of carries.

The architecture and operation of the first intermediate carry generator 104A will be described with reference to Figure 3 which is a schematic diagram of an example of an intermediate carry generator 104. The structure and operation of the other intermediate carry generators 104 will be analogous to that described with respect to the first intermediate carry generator 104A. The intermediate carry generators 104 will receive respective, merged propagate and generate signals from the third stage 208 of the sparse carry-merge circuit 102 (Figure 2) to generate additional carries or the missing 1 in 4 carries. The first intermediate carry generator 104A includes a first carry generate circuit 300, a second carry generate circuit 302 and a third carry generate circuit 304. Each of the carry generate circuits 300, 302 and 304 includes two rails or sub-circuits 306 and 308 to generate a pair of conditional carries in each circuit 300, 302 and 304. The first rails 306 of each circuit 300, 302 and 304 generate a conditional carry assuming a logic 0 carry-in signal to the intermediate carry generator 104A and the second rails 308 of each circuit 300, 302 and 304 generate an associated conditional carry assuming a logic 1 carry-in signal to the intermediate carry generator 104.

The intermediate carry generator 104A further includes 3 stages of ripple-carry logic formed by the first and second rails 306 and 308 of the circuits 300, 302 and 304. A first stage 318 of the first circuit 300 of the intermediate carry generator 104A will receive a propagate signal $P_{3:0}$ and an associated generate signal $G_{3:0}$ from the third stage 208 of the sparse carry-merge circuit 102. The propagate signal $P_{3:0}$ and the associated generate signal $G_{3:0}$ will be generated in the sparse carry-merge circuit 102 by merging bits 0-3 of the two binary number A_{63} and B_{63} as previously described. A first CM gate 320 in the first rail 306A of the first circuit 300 will merge the propagate signal $P_{3:0}$ and the associated generate signal $G_{3:0}$ with the assumed logic 0 carry-in signal in the first stage 318 to generate a resulting conditional carry $C_{3,0}$. A second CM gate 322 of the first circuit 300 in the second rail 308A will merge the propagate signals $P_{3:0}$ and the associated generate signal $G_{3:0}$ with the assumed logic 1 carry-in to generate a resulting conditional carry $C_{3,1}$ in the first stage 318 of the first circuit 300. The first CM gate 320 is coupled to a first inverter 324 in a second stage 326 of the first rail 306A of the intermediate carry generator 104A and the first inverter 324 is coupled to a second inverter 328 in a third stage 330 of the intermediate carry generator 104A. The second inverter 328 is coupled to a 2:1 multiplexer 332A. Similarly, the second CM gate 322 is coupled to a first inverter 334 of the second rail 308A in the second stage 326 and the first inverter 334 is coupled to a second inverter 336 of the second rail 308A in the third stage 330. The second inverter 336 is coupled to the multiplexer 332A. The multiplexer 332A receives a carry-in (C_{in} in Figure 1) and selects the first rail 306A or the second rail 308A to provide the missing 1 of 4 carries of the second predetermined number of carries. For the first carry generate circuit 300 of the intermediate carry generator 104a, the missing 1 of 4 carries is designated C_3 .

The second circuit 302 of the intermediate carry generator 104A receives merged propagate signal $P_{7:4}$ and merged generate signal $G_{7:4}$ from the third stage 208 of the sparse carry-merge circuit 102 (Figure2). The propagate signal $P_{7:4}$ and the generate signal $G_{7:4}$ are generated from bits 4-7 of the two binary number A_{63} and B_{63} in the same manner as that described above for signals $P_{3:0}$ and $G_{3:0}$. The signals $P_{7:4}$ and $G_{7:4}$ are inverted in the first stage 318 by an inverter 338 in the first rail 306B and an inverter

340 in the second rail 308B. In the second stage 326, the first rail 306B includes a first CM gate 342 and the second rail 308B includes a second CM gate 344. The first CM gate 342 is coupled to the first inverter 338 and to the first CM gate 320 of the first circuit 300. The first CM gate 342 merges the inverted $P_{7:4}$ and $G_{7:4}$ signals with the conditional carry $C_{3,0}$ from the first CM gate 320 of the first circuit 300 to generate a first conditional carry $C_{7,0}$. The second CM gate 344 merges the inverted $P_{7:4}$ and $G_{7:4}$ signals with the conditional carry $C_{3,1}$ from the second CM gate 322 of the first circuit 300 to generate a second conditional carry $C_{7,1}$.

The third stage 330 of the second circuit 302 includes an inverter 346 in the first rail 306B coupled to the first CM gate 342 and another inverter 348 in the second rail 308B coupled to the second CM gate 344. The outputs of both inverters 346 and 348 are coupled to a 2:1 multiplexer 332B. As with the first circuit 300, the multiplexer 332B selects between the conditional signals generated by the first rail 306B and the second rail 308B in response to one of the carry-in signals of the first predetermined number of carries or in response to one of the 1 in 16 carries from the sparse carry-merge circuit 102 to provide a second one of the missing 1 in 4 carries of the second predetermined number of carries. In the case of the second circuit 302 of the first intermediate carry generator 104A, the missing 1 in 4 carry is designated C_7 .

The third circuit 304 of the intermediate carry generator 104A receives merged propagate signal $P_{11:8}$ and merged generate signal $G_{11:8}$ from the third stage 208 of the sparse carry-merge circuit 102. The propagate signal $P_{11:8}$ and the generate signal $G_{11:8}$ are generated from bits 8-11 of the two binary numbers A_{63} and B_{63} in the same manner as that described above for signals $P_{3:0}$, $G_{3:0}$ and $P_{7:4}$, $G_{7:4}$. The first and second stages 320 and 326 of the first rail 306C and the second rail 308C of the third circuit 304 are inverters 352, 354, 356 and 358. The third stage 330 of the third circuit 304 includes a first CM gate 360 in the first rail 306C and a second CM gate 362 in the second rail 308C. The first CM gate 360 is coupled to the inverter 356 and to the first CM gate 342 in the second stage 326 of the second circuit 302. The second CM gate 362 is coupled to the inverter 358 and the second CM gate 344 of the second circuit 302. The first CM gate 360 merges the $P_{11:8}$ and $G_{11:8}$ signals with the conditional carry $C_{7,0}$ from the first

CM gate 342 of the second circuit 302 to generate a first conditional carry $C_{11,0}$. The second CM gate 362 merges the $P_{11,8}$ and $G_{11,8}$ signals with the conditional carry $C_{7,1}$ from the second CM gate 344 of the second circuit 302 to generate a second conditional carry $C_{11,1}$. The first CM gate 360 and the second CM gate 362 are each coupled to a 2:1 multiplexer 332C. Similar to circuits 300 and 302, the multiplexer 332C will select between the conditional signals generated by the first rail 306C and the second rail 308C to provide a third one of the second predetermined number of carries or a third one of the missing 1 in 4 carries. In the case of the third circuit 304 of the first intermediate carry generator 104A, the missing 1 in 4 carry is designated C_{11} . As previously discussed, the second, third and fourth intermediate carry generators 104B, 104C and 104D have a similar structure and will operate in the same manner as just described with respect to the first intermediate carry generator 104A. The intermediate carry generators 104A-D collectively generate the second predetermined number of carries or the missing 1 in 4 carries as illustrated in Figure 1. The intermediate carry generator 104A generates carries C_3 , C_7 and C_{11} as described above. Similarly to the intermediate carry generator 104A, the intermediate carry generator 104B generates carries C_{19} , C_{23} and C_{27} , the intermediate carry generator 104C generates carries C_{35} , C_{39} and C_{43} and the intermediate carry generator 104D generates carries C_{51} , C_{55} and C_{59} as shown in Figure 1.

As described with reference to Figure 3, the CM logic gates 320, 322, 342, 344, 360 and 362 are in a lower or later stage 318, 326 and 330 in each subsequent circuit 300, 302 and 304 to define the ripple carry-merge logic for generating the two rails of conditional 1 in 4 carries in each circuit 300, 302 and 304.

Figure 4 is a block schematic diagram of an example of a single conditional sum generator 106 of Figure 1. The sum generator 106 includes four circuits 400, 402, 404 and 406. Each circuit 400, 402, 404 and 406 includes a first rail 408 and a second rail 410 and has four stages 412, 414, 416 and 418. The sum generator 106 is also arranged in a ripple carry-merge structure similar to the intermediate carry gate 104A of Figure 3. The first stage 412 of the first circuit 400 includes an inverter 422 that is coupled to the first rail 408A and to the second rail 410A. The other stages 414, 416 and 418 of the

first circuit 400 also include inverters 424 in the first and second rails 408A and 410A. The inverter 422 of first stage 412 of the first circuit 400 will receive a first propagate signal P_i from the first stage 200 of the sparse carry-merge circuit 102 in Figure 2. The first rail 408A will then generate a first conditional sum, $sum_{i,0}$, and the second rail 410A will generate a second conditional sum, $sum_{i,1}$. The inverters 424 of the fourth stage 418 of each rail 408A and 410A are both coupled to a 2:1 multiplexer 426A. The multiplexer 426A will select between the conditional sums, $sum_{i,0}$ and $sum_{i,1}$, of the first rail 408A and the second rail 410A in response to one of the 1 in 4 carries or one of the predetermined number of combined first and second carries to provide a final sum, sum_i . For example, referring back to Figure 1, for the final sum_i where i equals zero (0) the carry-in will be C_{in} and the P_i input signal to the first circuit 400 will be P_0 from the first stage 200 of the sparse carry-merge circuit 102 (Figure 2).

The second circuit 402 includes a first combination CM/exclusive OR logic gate 428 in the second stage 414 of the first rail 408B and a second combination CM/exclusive OR logic gate 430 in the second stage 414 of the second rail 410B. The first stage 412, third stage 416 and fourth stage 418 of the first and second rails 408B and 410B of the second circuit 402 include inverters 432. The first combination CM/exclusive OR logic gate 428 will merge signals P_{i+1} and G_{i+1} with an assumed logic 0 carry-in to generate a conditional sum, $sum_{i+1,0}$. The second combination CM/exclusive OR gate 430 will merge signals P_{i+1} and G_{i+1} with an assumed logic 1 carry-in to generate a conditional sum, $sum_{i+1,1}$. The inverters 432 of the fourth stage 418 of the second circuit 402 are both coupled to a 2:1 multiplexer 426B. The multiplexer 426B selects between the two conditional sums, $sum_{i+1,0}$ and $sum_{i+1,1}$, generated by the first and second rails 408B and 410B of the second circuit 402 in response to one of the 1 in 4 carries to provide the final sum, sum_{i+1} .

The third circuit 404 and the fourth circuit 406 have similar structures to the first and second circuits 400 and 402 except the third circuit 404 has a first CM/exclusive OR gate 436 and a second CM/exclusive OR gate 438 in the third stage 416, and the fourth circuit 406 has a first CM/exclusive OR gate 440 and a second CM/exclusive OR gate 442 in the fourth stage 418. All of the other stages of the first and second rails 408

and 410 of the third and fourth circuits 404 and 406 include inverters 444 and 446, respectively. As with the second circuit 402, each of the CM/exclusive OR gates 436, 438, 440 and 442 are coupled to the inverter 444 and 446, respectively, of the preceding stage and to the CM/exclusive OR gate 428, 430, 436 or 438 in the corresponding first or second rail 408 or 410 of the preceding circuit. The fourth stage 418 of each of the third and fourth circuits 404 and 406 are coupled to respective 2:1 multiplexers 426C and 426D. The multiplexers 426C and 426D will then select between the conditional sums ($\text{sum}_{i+2,0}$ or $\text{sum}_{i+2,1}$ and $\text{sum}_{i+3,0}$ or $\text{sum}_{i+3,1}$) to provide the final sums, sum_{i+2} and sum_{i+3} , in response to the appropriate 1 in 4 carry as a function of the order of the bits being summed as shown in Figure 1. For example, for final sums, $\text{Sum}(63:60)$, the 1 in 4 carry or carry_{in} is C_{59} from Figure 1. For final sums, $\text{Sum}(31:28)$, the 1 in 4 carry is C_{27} and so forth for the other final sums.

Figure 5 is a block schematic diagram illustrating the critical path of the 64 bit quaternary-tree adder 100 of Figure 1. The number of transistors or gates in a summation path 500 from an input I/P to a SUMOUT for each stage of the sparse carry-merge circuit 102, the intermediate carry generator 104 and the sum generator 106 are shown by the numbers in each block representing each of the stages. For example, the number in the block of the first stage 200 of the sparse carry-merge circuit 102 is "3N" representing 3 N-channel transistors in the first stage 200. The critical path for the adder 100 of the present invention will be the sparse carry-merge circuit 102 and the multiplexers 332 and 426 because the 1 in 16 carries generated by the sparse carry-merge circuit 102 are needed first and are used by the multiplexers 332 and 426 to select between the conditional carries and the conditional sums in the intermediate carry generators 104 and the conditional sum generators 106. The number of transistors in the critical path provides a relative indication of the speed and efficiency of the adder 100 when compared to the critical paths of other more complex adders with more transistors in the critical path because each transistor or device in the critical path represents a certain amount of delay.

Figure 6 is a block schematic diagram of a 32 bit quaternary-tree adder 600 in accordance with the present invention. The adder 600 includes seven stages. The first

six stages forming a combination sparse carry-merge circuit 602 and an intermediate carry generator 604 shown by broken lines in Figure 6. The seventh stage includes a plurality of conditional sum generators 606A-H. The sparse carry-merge circuit 602 defines a first circuit adapted to generate a first group of a predetermined number of carries or 1 in 8 carries, C_7 , C_{15} and C_{23} and the intermediate carry generator 604 forms at least a portion of a second circuit or group of circuits defined below to generate a second predetermined number of carries or the additional missing 1 in 4 carries C_3 , C_{11} , C_{19} and C_{27} . The missing 1 in 4 carries C_{11} and C_{19} may each be generated using an inverter 608A and 608B and a two input NOR gate 610A and 610B.

As with the 64 bit adder 100 (Figure 1), the first stage 616 of the 32 bit adder 600 includes a plurality of logic gates 618 represented by the blocks in the first stage 616 to generate a plurality of propagate signals P_i (where $i=0$ to 32) and a plurality of generate signals G_i (where $i=0$ to 32). The logic gates 618 each perform an "exclusive OR" logic operation on each pair of the bits 0-32 of the binary numbers A_{32} and B_{32} to generate the plurality of propagate signals ($P_i=A_i \oplus B_i$) and an "AND" logic operation on each pair of bits of the two binary numbers A_{32} and B_{32} to generate the plurality of generate signals ($G_i=A_i \cdot B_i$). The second stage 620 includes a plurality of CM logic gates 622. Each CM logic gate 622 merges the propagate signals (P_i and P_{i-1}) and the generate signals (G_i and G_{i-1}) from two adjacent logic gates 618 of the first stage 616. The third stage 624 includes a plurality of CM logic gates 626. Each CM logic gate 626 merges the propagate signals and the generate signals from two adjacent logic gates 622 of the second stage 620. The 1 in 4 carry C_3 is then generated by passing the output generate signal G_{3-0} from CM logic gate 626a through inverters 628, 630 and 632 of the fourth stage 634, fifth stage 636 and sixth stage 638 respectively. The carry C_3 is then inputted to sum generator 606b.

The 1 in 8 carry C_7 is generated by merging C_{7-4} and C_3 in CM logic gate 640 in the fourth stage 634. The output of CM logic gate 640 is then passed through inverters 642 and 644 in the fifth and sixth stages 636 and 638 respectively to provide C_7 which is inputted to the sum generator 606C.

The carry C_{15} is generated by merging C_{15-12} and C_{11-8} in a CM logic gate 646 in the fourth stage 634 to provide merged carry C_{15-8} . The carry C_{15-8} is then merged with the carry C_{7-0} in CM logic gate 648 in the fifth stage 636. The output of CM logic gate 648 is then passed through an inverter 650 to provide the carry C_{15} which is inputted to sum generator 606E.

The carry C_{23} is generated by merging C_{23-20} from a CM logic gate 626F and C_{19-16} from a CM logic gate 626E, both in the third stage 624, in a CM logic gate 652 in the fourth stage 634. The output carry C_{23-16} is then merged with the carry C_{15-0} from the CM logic gate 648 in the fifth stage 636 in a CM logic gate 654 in the sixth stage 638 to provide the carry C_{23} which is inputted into the sum generator 606G.

The carry C_{27} is generated by using two additional carry-merge (CM) gates 656 and 658. The first CM gate 656 generates a merged carry C_{27-16} and the second CM gate 658 merges the carry C_{15} from the CM gate 648 and the carry C_{27-16} to generate the carry C_{27} which is inputted into the sum generator 606H.

Figure 7 is a schematic diagram of an example of a multiplexer recovery circuit 700 for use with the 64 bit adder circuit 100 or the 32 bit adder circuit 600 in accordance with the present invention. The multiplexer recovery circuit 700 is described with respect to the 64 bit adder 100 of Figure 1; although, the multiplexer recovery circuit 700 may be adapted for use with the 32 bit adder circuit 600 of Figure 6 as well or any adder that uses multiplexers to select between conditional values or states. The multiplexer recovery circuit 700 is coupled to each multiplexer 332 of each intermediate carry generator 104 (Figure 3) and each multiplexer 426 of each sum generator 106 (Figure 4) to correct for any erroneous discharge or change of state of the 1 in 16 carries or 1 in 4 carries used by the first and second multiplexers 332 and 426 respectively to select between the first or second input signals (or rails) to the multiplexers 332 and 426 or to correct the output sum if it should be erroneously discharged or altered. In Figure 7, a first input signal from a first rail 306 (Figure 3) would be coupled to a first input $MUX1INP_a$ of the multiplexer 332 and a second input signal from a second rail 308 (Figure 3) would be coupled to the a second input $MUX1INP_b$ of the multiplexer 332. The 1 in 16 carry_{in} from the sparse carry-merge

5 circuit 102 would be coupled to a MUXSELECT1 input of the multiplexer 332. The first input MUX1INP_a is coupled to a first transmission gate 702 of the multiplexer 332 and the second input MUX1INP_b is coupled to a second transmission gate 704 of the multiplexer 332. The MUXSELECT1 input is connected to a P-channel gate of the first transmission gate 702 and to an N-channel gate of the second transmission gate 704 and to an inverter 706. The output of the inverter 706 is connected to an N-channel gate of the first transmission gate 702 and to a P-channel gate of the second transmission gate 704. The outputs of the first and second transmission gates 702 and 704 are coupled to the input MUXSELECT2 of the second multiplexer 426. Accordingly, the first transmission gate 702 is activated when MUXSELECT1 is low to connect the first input MUX1INP_a to an output (MUXSELECT2) of the multiplexer 332 and the second transmission gate 704 is activated when MUXSELECT1 is high to connect the second input MUX1INP_b to the output of the first multiplexer 332. The outputs of the first and second transmission gates 702 and 704 are also coupled to a voltage supply 708 by a P-channel metal oxide semiconductor (PMOS) transistor 710. The operation of the PMOS transistor 710 is controlled by a clock, Clk, coupled to a gate of the PMOS transistor 710. A low clock pulse (Clk low) will activate the PMOS transistor 710 and pre-charge MUXSELECT2 to a high signal or high state. MUXSELECT2 is also coupleable to the supply voltage 708 by two series connect PMOS transistors 712 and 714. A gate of the PMOS transistor 712 is coupled to MUXSELECT1 and a gate of the PMOS transistor 714 is coupled to the input MUX1INP_a of the first transmission gate 702 by an inverter 716. Accordingly, MUXSELECT1 going low and MUX1INP_a going high will activate the PMOS transistors 712 and 714 to also pre-charge MUXSELECT2 or hold MUXSELECT2 high.

25 MUXSELECT2 is coupled to a P-channel gate of a first transmission gate 718 of the second multiplexer 426 and to an N-channel gate of a second transmission gate 720 of multiplexer 426. MUXSELECT2 is also coupled to an inverter 722 that couples MUXSELECT2 to an N-channel gate of the first transmission gate 718 and to a P-channel gate of the second transmission gate 720. Accordingly, when MUXSELECT2 is low, the first transmission gate 718 is activated to connect an input MUX2INP_a to the

output SUMOUT of the multiplexer recovery circuit 700 and when MUXSELECT2 is high, the second transmission gate 720 is activated to connect an input MUX2INP_b to the output SUMOUT of the multiplexer recovery circuit 700. The inputs MUX2INP_a and MUX2INP_b represent the first and second rails 408 and 410 respectively of one of the conditional sum generator circuits 400, 402, 404 or 406 of Figure 4. The output SUMOUT of the second multiplexer 426 is coupleable to the voltage supply 708 by a PMOS transistor 724. A gate of the PMOS transistor 724 is coupled to the clock, CLK. Accordingly, when the clock is low (CLK goes low) the transistor 724 will turn on to connect the voltage supply 708 to the output SUMOUT of the multiplexer 426. SUMOUT is also connectable to the supply voltage 708 by series connected PMOS transistors 726 and 728 or series connected PMOS transistors 730 and 732. A gate of the first PMOS transistor 726 is connected to the inverter 722 to control operation of the transistor 726, and a gate of the second PMOS transistor 728 is connected to an inverter 734 which in turn is connected to the second input MUX2INP_b of the second multiplexer 426. Thus, when MUXSELECT2 and MUX2INP_b are both high, the transistors 726 and 728 are turned on or activated to connect the output SUMOUT to the supply voltage 708.

A gate of the third PMOS transistor 730 is coupled to MUXSELECT2 to control operation of the PMOS transistor 730, and a gate of the fourth PMOS transistor 732 is coupled to the inverter 736 which in turn is connected to the first input MUX2INP_a. Therefore, when MUXSELECT2 is low and MUX2INP_a is high, SUMOUT is connected to the supply voltage 708 to pre-charge the output, SUMOUT.

In operation, CLK goes low or to a low state to activate the transistors 710 and 724 to cause MUXSELECT2 and SUMOUT to be pre-charged high or to a high state. The CLK will go high to turn off the transistors 710 and 724 to begin an evaluation of the inputs MUX1INP_a and MUX1INP_b to the first multiplexer 332 and MUX2INP_a and MUX2INP_b to the second multiplexer 426. As soon as the CLK goes high, MUXSELECT2 having been pre-charged high will activate the second transmission gate 720 of the second multiplexer 426 and SUMOUT will assume the value of MUX2INP_b. This may or may not be the correct value or state for SUMOUT because

the correct value of MUXSELECT2 will only be resolved after MUXSELECT1 is determined and the correct or true input MUX1INP_a or MUS1INP_b to the first multiplexer 332 is selected to be coupled to the output (MUXSELECT2). The problem is that the CLK can go high before MUXSELECT1 is determined or correctly resolved. Depending upon the value or state of MUXSELECT1 at the instant the CLK goes high or at any time before MUXSELECT1 is properly resolved, MUXSELECT2 may be coupled to the incorrect input, either MUX1INP_a or MUX1INP_b, and therefore evaluated or resolved incorrectly. To insure that MUXSELECT2 is resolved correctly, The PMOS transistors 712 and 714 will detect the input signals MUXSELECT1 and MUX1INP_a coming into the first multiplexer 332 before the transmission gates 702 and 704 resolve the correct value or state for MUXSELECT2. The PMOS transistors 712 and 714 will either be activated to bring MUXSELECT2 to a high state or will remain off so that MUXSELECT2 remains at the value or state it becomes after CLK goes high.

Similarly, the instant CLK goes high, SUMOUT will assume the value or state of MUX2INP_b because MUXSELECT2 is pre-charge high by the PMOS transistor 710 and the second transmission gate 720 of the second multiplexer 426 will be active when CLK goes high to turn off PMOS transistor 710. MUX2INP_b may or may not be the correct value or evaluation for SUMOUT since MUXSELECT2 has not been properly evaluated or determined. Accordingly, after MUXSELECT1 has been properly resolved and MUXSELECT2 properly determined, the PMOS transistors 726 and 730 detect the proper value or true evaluation of MUXSELECT2; the PMOS transistor 728 detects the value or state of the input MUX2INP_b and the PMOS transistor 732 detects the value or state of the input MUX2INP_a before the transmission gates 718 and 720 properly resolve the true or correct value for SUMOUT. Depending upon the states detected, either the PMOS transistors 726 and 728 or the PMOS transistors 730 or 732 will be activated to pre-charge the output SUMOUT or the transistors 726, 728, 730 and 732 will remain inactive so that SUMOUT remains at the same state or value it assumes between the time CLK goes high and when SUMOUT is properly evaluated or resolved. Accordingly, the multiplexer recovery circuit 700 insures that the output sum or

SUMOUT from each of the conditional sum generators 106 is correctly resolved. The multiplexer recovery circuit 700 enables a single-rail dynamic implementation of the adder core 100.

Figure 8 is a flow chart of a method 800 of operation of a quaternary-tree adder 100 in accordance with the present invention. In step 802, a first predetermined number of carries, such as 1 in 16 carries for a 64 bit adder 100 or 1 in 8 carries for a 32 bit adder, are generated by merging bits of two binary number. In step 804, a plurality of conditional carries are generated assuming a logic 0 carry-in signal and a second plurality of conditional carries are generated assuming a logic 1 carry-in signal. A selection is made between each of the plurality of conditional carries generated assuming a logic 0 carry-in and an associated or corresponding conditional carry assuming a logic 1 carry-in in step 808. The selection is made in response to or using one of the first predetermined number of carries to provide a second predetermined number of carries, such as missing carries from the first predetermined number of carries to provide 1 in 4 carries or 1 carry for every group of 4 bits of the two binary numbers being summed, grouping from a least significant digit. In step 810, a plurality of conditional sums are generated assuming a logic 0 carry-in and a second plurality of conditional sums are generated assuming a logic 1 carry-in. In step 812, a selection is made between each one of the conditional sums for the logic 0 carry-in and an associated one of the plurality of conditional sums for a logic 1 carry-in to provide a final sum of the two binary numbers. The selection is made in response to or using a carry-in from the first and second predetermined numbers of carries, such as the 1 in 4 carries which also includes the 1 in 16 carries for a 64 bit adder or 1 in 8 carries for a 32 bit adder.

Figure 9 is a block schematic diagram of an example of an electronic system 900, such as a computer system or the like, that incorporates the 64 bit quaternary-tree adder 100 (Figure 1) or the 32 bit adder 600 (Figure 6) in accordance with the present invention. The electronic system 900 includes a processor 902, central processing unit (CPU) or the like. The processor 902 includes at least one arithmetic logic unit (ALU) 904 to perform calculations on integer numbers. To perform these arithmetic

calculations, the ALU 904 will contain a plurality of 64 bit adders 100 (Figure 1) or 32 bit adders 600 (Figure 6). It will be appreciated that the processor 902 will also include other components 906, such as a floating point math unit for calculations involving floating point number, cache memories, buffer interface units, buffers and fetch/decode units to name of few examples of other components 906. The processor 902 is coupled to at least one memory system 908 to store programs and other information. The processor 902 and memory system 908 may be contained in a housing or computer unit 910. Depending upon the function and features of the electronic or computer system 900, other components (not shown) may be contained in the unit 902.

The electronic or computer system 900 will also usually include user interface equipment 912 coupled to the processor 902 and memory system 908 to control the operation of the system 900. Examples of user interface equipment 912 depicted in Figure 9 are a monitor 914, a keyboard 916, a printer 918, and a pointing device 918 or mouse.

Although specific embodiments have been illustrated and described herein, it will be appreciated by those of ordinary skill in the art that any arrangement which is calculated to achieve the same purpose may be substituted for the specific embodiment shown. This application is intended to cover any adaptations or variations of the present invention. Therefore, it is intended that this invention be limited only by the claims and the equivalents thereof.